AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions and listing of claims in the application:

40. (Currently Amended) A method for determining a constraint used for ensuring type safe linkage, comprising:

creating an entry in a constraint table for a class name;

creating a first entry in a loaded class cache (LCC) for the class name and a first loading object;

creating a second entry in the LCC for the class name and a second loading object; and

defining the constraint table entry based on the first and second entries in the LCC to ensure type safe linkage.

41. (Previously Presented) The method of claim 40, wherein defining the constraint table entry comprises:

indicating an error based on a determination that a class type associated with the first entry is not compatible with a class type associated with the second entry.

42. (Previously Presented) The method of claim 40, wherein defining the constraint table entry comprises:

defining a pair of objects for the constraint table entry that includes a set comprising the first and second loading objects and an indication reflecting either a

class type associated with the first entry or a class type associated with the second entry.

43. (Previously Presented) The method of claim 42, wherein the pair of objects is defined based on a determination that the first and second loading objects are not associated with a set of loading objects corresponding to the constraint table entry.

44. (Previously Presented) The method of claim 40, wherein defining the constraint table entry comprises:

determining that a first set corresponding to the constraint table entry includes the first loading object and not the second loading object.

45. (Previously Presented) The method of claim 44, further comprising:

adding the second loading object to the first set; and

defining a pair of objects corresponding to the constraint table entry that includes the first set and either a class type associated with the first entry or a class type associated with the second entry.

46. (Previously Presented) The method of claim 40, wherein defining the constraint table entry comprises:

determining that a first set corresponding to the constraint table entry includes the second loading object and not the first loading object.

47. (Previously Presented) The method of claim 46, further comprising:

adding the first loading object to the first set; and

defining a pair of objects corresponding to the constraint table entry that includes the first set and either a class type associated with the first entry or a class type associated with the second entry.

48. (Previously Presented) The method of claim 40, wherein defining the constraint table entry comprises:

determining that a first set corresponding to the constraint table entry includes the first loading object and a second set corresponding to the constraint table entry includes the second loading object.

49. (Previously Presented) The method of claim 48, further comprising:

merging the first set and the second set into a new set; and

defining a pair of objects corresponding to the constraint table entry that includes the new set and either a class type associated with the first entry or a class type associated with the second entry.

50. (Previously Presented) A method for providing type safe linkage, comprising:

providing a cache that maps a class name and loader object to a class type;

providing a constraint table that maps the class name to one or more pairs of a class type and a set of loader objects; and

providing type safe linkage during execution of a process based on the cache and constraint table.

51. (Previously Presented) The method of claim 50, wherein providing a cache comprises:

creating an entry in the cache that returns a class type equal to the class type included in the one or more pairs based on the class name and loader object.

52. (Previously Presented) The method of claim 51, wherein creating an entry in the cache comprises:

creating an entry in the constraint table that is indexed by the class name; and

setting the class type equal to the class type included in the one or more pairs based on a determination that the loader object is not included in the set of loader objects.

53. (Previously Presented) The method of claim 51, wherein creating an entry in the cache comprises:

determining that the loader object is included in the set of loader objects; and

setting the class type equal to the class type included in the one or more pairs based on a determination whether the class type and the class type included in the one or more pairs are compatible.

54. (Previously Presented) The method of claim 50, wherein providing type safe linkage during execution of a process comprises:

determining that the class name is referenced;

determining, in response to the reference, whether the class was previously loaded by the loader object based on the cache; and

loading the referenced class name using the loader object and placing an entry for the loaded class name into the cache.

55. (Previously Presented) The method of claim 52, wherein creating an entry in the constraint table that is indexed by the class name is based on a determination that there is no entry in the constraint table for the class name.

56. (Previously Presented) The method of claim 53, wherein the class types are compatible when they are the same type or either class type is a null type.

57. (Currently Amended) A system for ensuring type safe linkage during execution of a program, comprising:

a memory including instructions ~~reflecting~~ for performing a process for identifying a first class that makes a symbolic reference to an attribute contained in a second class, instructions ~~reflecting~~ for performing a process for imposing a constraint associated with the referenced attribute, and instructions ~~reflecting~~ for performing a process for verifying when the program is executed that the symbolic reference complies with the constraint; and

a processor for executing the instructions included in the memory to ensure type safe linkage during execution of the program.

58. (Previously Presented) The system of claim 57, wherein the constraint requires that a type of the attribute, when loaded by a loader that defines the first class, is the same as the type when loaded by a loader that defines the second class.

59. (Previously Presented) The system of claim 57, wherein the attribute is a field that is contained in the second class.

60. (Previously Presented) The system of claim 57, wherein the attribute is a method that is contained in the second class.

61. (Previously Presented) The system of claim 57, wherein the instructions reflecting a process for verifying is executed by the processor when the attribute is loaded by a loader that defines at least one of the first and second classes.

62. (Currently Amended) A system for determining a constraint used for ensuring type safe linkage, comprising:

means for creating an entry in a constraint table for a class name;

means for creating a first entry in a loaded class cache (LCC) for the class name and a first loading object;

means for creating a second entry in the LCC for the class name and a second loading object; and

means for defining the constraint table entry based on the first and second entries in the LCC to ensure type safe linkage.

63. (Previously Presented) The system of claim 62, wherein the means for defining the constraint table entry comprises:

means for indicating an error based on a determination that a class type associated with the first entry is not compatible with a class type associated with the second entry.

64. (Previously Presented) The system of claim 62, wherein the means for defining the constraint table entry comprises:

means for defining a pair of objects for the constraint table entry that includes a set comprising the first and second loading objects and an indication reflecting either a class type associated with the first entry or a class type associated with the second entry.

65. (Previously Presented) The system of claim 64, wherein the pair of objects is defined based on a determination that the first and second loading objects are not associated with a set of loading objects corresponding to the constraint table entry.

66. (Previously Presented) The system of claim 62, wherein the means for defining the constraint table entry comprises:

means for determining that a first set corresponding to the constraint table entry includes the first loading object and not the second loading object.

67. (Previously Presented) The system of claim 66, further comprising:

means for adding the second loading object to the first set; and

means for defining a pair of objects corresponding to the constraint table entry that includes the first set and either a class type associated with the first entry or a class type associated with the second entry.

68. (Previously Presented) The system of claim 62, wherein the means for defining the constraint table entry comprises:

means for determining that a first set corresponding to the constraint table entry includes the second loading object and not the first loading object.

69. (Previously Presented) The system of claim 68, further comprising:

means for adding the first loading object to the first set; and

means for defining a pair of objects corresponding to the constraint table entry that includes the first set and either a class type associated with the first entry or a class type associated with the second entry.

70. (Previously Presented) The system of claim 62, wherein the means for defining the constraint table entry comprises:

means for determining that a first set corresponding to the constraint table entry includes the first loading object and a second set corresponding to the constraint table entry includes the second loading object.

71. (Previously Presented) The system of claim 70, further comprising:

means for merging the first set and the second set into a new set; and

means for defining a pair of objects corresponding to the constraint table entry that includes the new set and either a class type associated with the first entry or a class type associated with the second entry.

72. (Previously Presented) A system for providing type safe linkage, comprising:

means for providing a cache that maps a class name and loader object to a class type;

means for providing a constraint table that maps the class name to one or more pairs of a class type and a set of loader objects; and

means for providing type safe linkage during execution of a process based on the cache and constraint table.

73. (Previously Presented) The system of claim 72, wherein the means for providing a cache comprises:

means for creating an entry in the cache that returns a class type equal to the class type included in the one or more pairs based on the class name and loader object.

74. (Previously Presented) The system of claim 73, wherein the means for creating an entry in the cache comprises:

means for creating an entry in the constraint table that is indexed by the class name; and

means for setting the class type equal to the class type included in the one or more pairs based on a determination that the loader object is not included in the set of loader objects.

75. (Previously Presented) The system of claim 73, wherein the means for creating an entry in the cache comprises:

means for determining that the loader object is included in the set of loader objects; and

means for setting the class type equal to the class type included in the one or more pairs based on a determination whether the class type and the class type included in the one or more pairs are compatible.

76. (Previously Presented) The system of claim 72, wherein the means for providing type safe linkage during execution of a process comprises:

means for determining that the class name is referenced;

means for determining, in response to the reference, whether the class was previously loaded by the loader object based on the cache; and

means for loading the referenced class name using the loader object and placing an entry for the loaded class name into the cache.

77. (Previously Presented) The system of claim 74, wherein the means creating an entry in the constraint table that is indexed by the class name includes means for determining whether there is an entry in the constraint table for the class name.

78. (Previously Presented) The system of claim 75, wherein the class types are compatible when they are the same type or either class type is a null type.

79. (Currently Amended) A computer-readable storage medium including instructions for performing a method, when executed by a processor, for determining a constraint used for ensuring type safe linkage, the method comprising:

creating an entry in a constraint table for a class name;

creating a first entry in a loaded class cache (LCC) for the class name and a first loading object;

creating a second entry in the LCC for the class name and a second loading object; and

defining the constraint table entry based on the first and second entries in the LCC to ensure type safe linkage.

80. (Currently Amended) The computer-readable storage medium of claim 79, wherein defining the constraint table entry comprises:

indicating an error based on a determination that a class type associated with the first entry is not compatible with a class type associated with the second entry.

81. (Currently Amended) The computer-readable storage medium of claim 79, wherein defining the constraint table entry comprises:

defining a pair of objects for the constraint table entry that includes a set comprising the first and second loading objects and an indication reflecting either a class type associated with the first entry or a class type associated with the second entry.

82. (Currently Amended) The computer-readable storage medium of claim 81, wherein the pair of objects is defined based on a determination that the first and second loading objects are not associated with a set of loading objects corresponding to the constraint table entry.

83. (Currently Amended) The computer-readable storage medium of claim 79, wherein defining the constraint table entry comprises:

determining that a first set corresponding to the constraint table entry includes the first loading object and not the second loading object.

84. (Currently Amended) The computer-readable storage medium of claim 83, further comprising:

adding the second loading object to the first set; and

defining a pair of objects corresponding to the constraint table entry that includes the first set and either a class type associated with the first entry or a class type associated with the second entry.

85. (Currently Amended) The computer-readable storage medium of claim 79, wherein defining the constraint table entry comprises:

determining that a first set corresponding to the constraint table entry includes the second loading object and not the first loading object.

86. (Currently Amended) The computer-readable storage medium of claim 85, further comprising:

adding the first loading object to the first set; and

defining a pair of objects corresponding to the constraint table entry that includes the first set and either a class type associated with the first entry or a class type associated with the second entry.

87. (Currently Amended) The computer-readable storage medium of claim 79, wherein defining the constraint table entry comprises:

determining that a first set corresponding to the constraint table entry includes the first loading object and a second set corresponding to the constraint table entry includes the second loading object.

88. (Currently Amended) The computer-readable storage medium of claim 87, further comprising:

merging the first set and the second set into a new set; and

defining a pair of objects corresponding to the constraint table entry that includes the new set and either a class type associated with the first entry or a class type associated with the second entry.

89. (Currently Amended) A computer-readable storage medium including instructions for performing a method, when executed by a processor, for providing type safe linkage, the method comprising:

providing a cache that maps a class name and loader object to a class type;

providing a constraint table that maps the class name to one or more pairs of a class type and a set of loader objects; and

providing type safe linkage during execution of a process based on the cache and constraint table.

90. (Currently Amended) The computer-readable storage medium of claim 89, wherein providing a cache comprises:

creating an entry in the cache that returns a class type equal to the class type included in the one or more pairs based on the class name and loader object.

91. (Currently Amended) The computer-readable storage medium of claim 90, wherein creating an entry in the cache comprises:

creating an entry in the constraint table that is indexed by the class name; and

setting the class type equal to the class type included in the one or more pairs based on a determination that the loader object is not included in the set of loader objects.

92. (Currently Amended) The computer-readable storage medium of claim 90, wherein creating an entry in the cache comprises:

determining that the loader object is included in the set of loader objects; and

setting the class type equal to the class type included in the one or more pairs based on a determination whether the class type and the class type included in the one or more pairs are compatible.

93. (Currently Amended) The computer-readable storage medium of claim 89, wherein providing type safe linkage during execution of a process comprises:

determining that the class name is referenced;

determining, in response to the reference, whether the class was previously loaded by the loader object based on the cache; and

loading the referenced class name using the loader object and placing an entry for the loaded class name into the cache.

94. (Currently Amended) The computer-readable storage medium of claim 91, wherein creating an entry in the constraint table that is indexed by the class name is based on a determination that there is no entry in the constraint table for the class name.

95. (Currently Amended) The computer-readable storage medium of claim 92, wherein the class types are compatible when they are the same type or either class type is a null type.